考虑排错过程引进故障的开源软件可靠性模型研究 *

米晓萍，王金勇†

(山西大学 软件学院，太原 030006)

摘　要：近年来，开源软件在软件行业很受欢迎。但是，开源软件的可靠性却受到人们的广泛质疑。如何评估开源软件的可靠性是一个重要的问题。与传统的闭源软件相比，在建立开源软件可靠性模型时，必须考虑故障引入和故障检测与排错之间的延迟时间这两个因素。考虑了排错过程和不完美调试现象，提出了相应的开源软件可靠性模型。并且用两个开源软件故障数据集实来验证提出模型的拟合性能与预测性能。实验结果表明，提出的模型在开源软件可靠性评估中具有良好的拟合和预测性能。提出的模型可以用于开源软件在实际的开发过程中的可靠性评估。

关键词：软件可靠性；软件可靠性模型；排错过程；不完美调试；开源软件

中图分类号：TP311　　doi: 10.3969/j.issn.1001-3695.2018.04.0271

Software reliability models for open source software considering correction process and fault introduction

Mi Xiaoping, Wang Jinyong†

(*School of Software Engineering Shanxi University, Taiyuan* 030006, *China*)

Abstract: Recently, open source software is popular in software industry. However, the reliability for open source software is widely questioned by people. How to assess the reliability of open source software is an important issue. Compared with traditional closed source software, two factors, i. e. fault introduction and the delay time between the fault detection and correction are necessarily considered when building software reliability model of open source software. In this paper, we propose software reliability models for open source software considering the correction process and imperfect debugging. We use the experiment to validate the goodness-of-fit and predictive performance of the proposed model using two fault data sets of open source software. The experimental results show that the proposed models have a good fitting and predictive power in the reliability evaluation of open source software. The proposed models can be used to evaluate the reliability of open source software in the real-world development of open source software.

Key words: software reliability; software reliability model; correction process; imperfect debugging; open source software.

0　引言

在现代信息社会中，软件的应用越来越广泛。不仅软件的编码规模和功能越来越大，而且其开发方式也发生了很大的变化。例如，近年来流行的开源软件开发方法。开源软件的发展不同于传统的闭源软件开发。Raymond[1]把开源软件的开发模式称为 Bazaar（集市），把传统的闭源软件开发称为 Cathedral（大教堂）。此外，开源软件的测试由开发人员、社区志愿者和用户完成。由于开源软件开发具有开放性和动态性特点，它的可靠性受到了广泛质疑[2]。

一般来说，可以使用软件可靠性增长模型(SRGM)来评估软件可靠性和预测软件中剩余故障的数量。对于传统的闭源软件来说，在过去的四十年里已经开发了许多软件可靠性模型。他们大多只考虑故障检测过程建立软件可靠性模型[3-10]。例如，a）完美调试模型，Goel 和 Okumoto[3]认为在软件测试过程中失效率和剩余的故障数成正比例，提出了一种基于非齐次泊松过程的软件可靠性增长模型（NHPP）；b）不完美的调试模式，Goel[6]是第一个提出了一个不完美的调试模型。

另一方面，对于闭源软件，通常假设软件测试过程中检测到故障立即被去除。换言之，检测到的故障数与排错（修正）故障数相同。这种假设显然与实际的软件测试不一致。因为实际的故障检测和排错之间存在时间延迟。因此，Schneidewind[11]认为排错故障跟检测故障之间不同步。否则，会低估了软件中剩余故障的数量。他提出了一种考虑故障修正过程的时间延迟变量建模方法。Xie 和 Zhao[12]修改 Schneidewind 模型，并提出时间延迟作为增函数的软件可靠性模型。随后，研究者提出了

收稿日期：2018-04-20；修回日期：2018-07-06　　基金项目：NSFC-广东联合基金（第二期）超级计算科学应用研究专项基金资助项目（U1501501）；山西省自然科学基金资助项目（201601D011046）；国家自然科学基金资助项目（61702315）

作者简介：米晓萍（1976-），女，山西平遥人，副教授，硕士，主要研究方向为软件可靠性和故障预测、物联网；王金勇（1974-），男（通信作者），黑龙江鸡西人，讲师，博士研究生，主要研究方向为软件可靠性和故障预测（wangjinyong818@163.com）.

一系列与时间延迟有关的闭源软件可靠性模型[13-17]。

在开源软件可靠性建模方面，Tamura 和 Yamada[18]提出一种基于随机微分方程的软件可靠性模型，但只考虑故障检测过程。Zou 和 Davis[19]用几种传统的经典闭源软件可靠性模型对开源软件进行可靠性评估。实验结果表明，传统的闭源软件可靠性模型可以用来评估开源软件的可靠性。特别是，基于威布尔分布的软件可靠性模型在评估开源软件可靠性方面具有更好的拟合和预测性能。Li 等人[20]考虑开源软件开发过程中志愿者的兴趣随测试时间的变化情况，提出一种基于故障检测率随测试时间有先增后减变化的开源软件可靠性模型。

一般来说，开源软件的开发环境要比闭源软件复杂得多。开源软件的开发和测试相关人员比闭源软件更具多样性。开源软件的管理比闭源软件更松散。开源软件的开发过程比闭源软件更具动态性。因此，使用闭源软件可靠性模型来评估开源软件的可靠性，这与实际的软件测试环境是不完全一致的。虽然一些开源软件的可靠性模型已经建立，可以用来评估开源软件在某些软件测试情况下的可靠性，但大多数都是基于故障检测建立的，没有考虑故障排错和故障检测之间的时间延迟情况。而实际上在开源软件测试过程中，存在着故障检测和排错之间的时间延迟现象。例如，在 Apache 开源软件项目中，检测到的错误将被标记为创建、更新和解决时间等。检测到的故障状态可以开放、重新开放、解决和关闭等。因此，很明显看到开源软件中检测故障与排错故障之间存在时间延迟问题。另外，由于故障已被标记为已解决或已关闭，可以在以后重新被开放，说明在排除该故障时，原故障没有被完全消除或引入新故障。因此，在开源软件测试过程中，在建立开源软件可靠性模型时，还需要全面考虑故障引入现象。

本文提出了考虑开源软件测试过程中的故障排错过程和故障引入现象的软件可靠性模型。用最小二乘估计（LSE）估计模型的参数。并且使用开源软件的两个故障数据集验证了模型的性能。实验结果表明，考虑故障排错过程和故障引入现象的软件可靠性模型具有良好的故障拟合和预测性能。

本文的贡献如下，a)首先提出了考虑故障排错和故障引入的开源软件可靠性模型；b)故障检测与排错之间的时间延迟和故障引入现象是建立开源软件可靠性模型的两个重要因素；c)在开源软件的实际开发过程中，提出的模型可以被用来评估开源软件的可靠性。

本文使用的符号说明如下：

符号	说明	符号	说明
a	期望软件中最初存在故障的数量	a(t)	故障内容（总个数）函数
b	故障检测率	N(t_i)	故障检测出的数量
c	故障排错（修正）率	m_{t_i}	实际故障检测出的数量
α	故障引进率	n_{t_i}	实际故障排错（修正）的数量
k	样本大数量		

1 相关工作

在这一节中，介绍在软件测试过程中考虑故障排错已有的相关软件可靠性模型。在故障检测和故障排错方面，有关文献已进行了一些研究。例如，在闭源软件软件可靠性建模方面，Schniedewind[11]把故障检测和排错之间的时间延迟作为一个随机变量并且服从指数分布，提出了一种考虑故障修正过程的软件可靠性模型。Xie 和 Zhao[12]认为，故障检测和修正之间的时间延迟随着时间的推移逐渐增加，并提出了一种考虑时间延迟作为增函数的改进的软件可靠性模型。Lo 和 Huang[13]认为在软件测试过程中，检测到的故障立即被排除是一个不切实际的假设，提出将故障检测和修正过程综合考虑来建立相应的软件可靠性模型。Huang 和 Lin[14]也提出了一个考虑故障依赖和故障检测与修正之间的时间延迟的软件可靠性模型。Wu 等人[15]考虑了故障的依赖性性，并建立了另一种考虑故障检测和修正过程时间延迟的软件可靠性模型。Xie 等人[16]考虑了故障检测和排错过程之间的存在时间延迟现象，提出了相应的软件可靠性模型，并给出了考虑故障检测和排错过程的最优软件发布时间的成本模型。Peng 等人[21]通过整合测试工作量和故障引入现象，提出了一种闭源软件可靠性模型。

最近，Liu 等人[17]提出了一种基于马尔可夫的软件可靠性评估方法，并采用加权最小二乘估计方法对模型参数进行估计。Liu 等人[22]认为故障检测和排错过程之间的时间延迟是服从指数或威布尔分布的随机变量，并提出了相应的软件可靠性模型。他们还建立了多版本软件可靠性模型，并使用开源软件故障数据集来验证相关模型的性能。此外，Yang 等人[23]假设故障检测和排错过程之间的时间延迟服从伽马（Gamma）分布，并建立了多版本软件可靠性模型。他们还使用了两个开源软件故障数据集来验证模型的性能。由于开源软件的开发和测试环境的复杂性，建立开源软件的可靠性模型是非常困难的。Ullah 等人[24]提出了一种优化选择模型的方法，并对已建立的软件可靠性模型进行优化选择。最终选择出最优的模型来进行开源软件可靠性评估。

虽然已建立地开源软件可靠性模型考虑了故障检测和故障排错之间存在延迟的问题，但在实际的开源软件开发和测试过程中，由于没有考虑故障去除过程可能引进故障的情况，因此，用于建立地时间延迟的开源软件可靠性模型的假设并不符合实际的开源软件去除故障的情况。它们假设的合理性受到质疑。在本文中，提出一种不完美排错的开源软件可靠性模型，它是在充分考虑实际的开源软件故障检测和故障去除之间存在时间延迟的基础上，并且考虑去除故障时可能引进故障的情况来建立相应的开源软件可靠性模型。由于考虑了在开源软件故障去除时，可能引进故障的情况，因此，建立地开源软件可靠性模型更符合实际的故障检测和故障去除变化的过程，建立地开源软件可靠性模型的假设更加的合理和可信。

2 不完美排错的开源软件可靠性模型

2.1 基于 NHPP 的一般软件可靠性模型

假设{N(t), t≥0}是一个计数过程，表示到 t 时刻为止，累积检测故障的数量，并服从非齐次泊松分布（NHPP），那么基于 NHPP 的软件可靠性增长模型的均值函数（MVF）可以表示如下，

$$\Pr\{N(t)=n\}=\frac{(m(t))^n}{n!}\exp(-m(t)) \quad (1)$$

一般来说，大多数的基于 NHPP 的软件可靠性增长模型是假设在软件测试过程中瞬间检测到的故障数量和软件中剩余的故障数量成正比，如 G-O 模型[3]，延迟的 S 形模型（delayed S-shaped）[4]和拐点 S 形模型（Inflected S-shaped）[5]。可以用下面的方程表示，

$$m_d(t+\Delta t)\text{-}m_d(t)=b(t)(a(t)-m_d(t))+o(\Delta t) \quad (2)$$

$$\frac{dm_d(t)}{dt}=b(t)(a(t)-m_d(t)) \quad (3)$$

在方程（3）中，当 a(t)=a 和 b(t)=b, 那么解方程可得 $m_d(t)=a(1-\exp(-bt))$, 它是 G-O 模型；当 a(t)=a 和 $b(t)=\frac{b^2t}{1+bt}$，解方程可得，$m_d(t)=a(1-(1+bt)\exp(-bt))$；当 a(t)=a 和 $b(t)=\frac{b}{1+\beta\exp(-bt)}$，解方程可得，$m_d(t)=\frac{a(1-\exp(-bt))}{1+\beta\exp(-bt)}$。

由于在软件测试过程中存在故障检测和排错之间的延迟时间，Schneidewind[11]提出了一种考虑故障修正和故障检测非同步的时间延迟模型。它可以如下所示，

$$m_c(t)=m_d(t-\Delta t) \quad (4)$$

上式中，Δt 是一个延迟时间变量。Xie 和 Zhao[12]修改 schneidwind 的模型，他们提出的其他形式，例如，在（$t, t+\Delta t$）时间区间，排错率和检测出故障但未去除的故障数量正比。它可以表示如下，

$$\frac{dm_c(t)}{dt}=c(t)(m_d(t)-m_c(t)) \quad (5)$$

2.2 整合不完美调试和排错过程的模型

提出模型假设：

（1）故障检测过程与故障排错过程服从非齐次泊松过程（NHPP）[12,23]。

（2）瞬时检测到故障的数量与软件中剩余故障的数量成正比[12]。

（3）瞬时修正后的故障数量与软件检测到的故障但未去除的故障数量成正比[12]。

（4）检测出的故障不会立即被去除，在排错时可能以概率 α 引入新的故障。

从上述假设出发，可以建立如下微分方程，

$$\begin{cases}\frac{dm_d(t)}{dt}=b(t)(a(t)-m_d(t)) & (6)\\ \frac{dm_c(t)}{dt}=c(t)(m_d(t)-m_c(t)) & (7)\\ \frac{da(t)}{dt}=\alpha\frac{dm_d(t)}{dt} & (8)\end{cases}$$

已知条件 a(0)=a，方程（8）可以得出下式，

$$a(t)=a(1+\alpha\, m(t)) \quad (9)$$

当 b(t)=b， 把等式（9）代入等式（6），可以得出下式，

$$m_d(t)=\frac{a}{1-\alpha}(1-\exp(-b(1-\alpha)t)) \quad (10)$$

当 c(t)=c，把等式（9）代入等式（7），可以得出下式，

$$m_c(t)=\frac{ac}{1-\alpha}\left(\frac{1}{c}-\frac{\exp(-ct)}{c}-\frac{\exp(-b(1-\alpha)t)-\exp(-ct)}{c-b(1-\alpha)}\right) \quad (11)$$

因此，故障检测和不完美排错过程的均值函数可以如下所示，

$$\begin{cases}m_d(t)=\frac{a}{1-\alpha}(1-\exp(-b(1-\alpha)t))\\ m_c(t)=\frac{ac}{1-\alpha}\left(\frac{1}{c}-\frac{\exp(-ct)}{c}-\frac{\exp(-b(1-\alpha)t)-\exp(-ct)}{c-b(1-\alpha)}\right)\end{cases} \quad (12)$$

2.3 提出的模型的推导过程

1）把公式（9）代入公式（6），那么方程（6）可以被表示为一下形式，

$$\frac{dm_d(t)}{a-(1-\alpha)m_d(t)}=b(t)dt \quad (13)$$

上式两边积分，

$$\int d(\ln(a-(1-\alpha)m_d(t)))=-(1-\alpha)\int bdt$$

$$\ln(a-(1-\alpha)m_d(t))=\int -(1-\alpha)bdt$$

$$a-(1-\alpha)m_d(t)=C_1\exp(-(1-\alpha)bt)$$

其中 C_1 为常量。当 t=0， $m_d(0)=0$. 因此，

$$a-(1-\alpha)m_d(t)=a\exp(-(1-\alpha)bt)$$

$$m_d(t)=\frac{a}{1-\alpha}(1-\exp(-b(1-\alpha)t)) \quad (14)$$

2)设 C(t)= $\int_0^t cdx$ =ct, 那么公式(7)可以改写成以下形式,

$$\frac{dm_c(t)}{dt}+C(t)m_c(t)=C(t)m_d(t) \quad (15)$$

$$\exp(C(t))dm_c(t)+\exp(C(t))C(t)m_c(t)dt=\exp(C(t))C(t)m_d(t)dt$$

两边积分得，

$$\int d(\exp(C(t))m_c(t))=\int \exp(C(t))C(t)m_d(t)dt$$

$$\exp(C(t))m_c(t)=\int_0^t \exp(C(t))C(t)m_d(t)dt \quad (16)$$

把公式（9）代入公式（16），

$$m_c(t)=\exp(-ct)\int_0^t \exp(ct)cm_d(t)dt$$

$$m_c(t)=\exp(-ct)\int_0^0 \exp(ct)c\frac{a}{1-\alpha}(1-\exp(-b(1-\alpha)t))dt$$
$$=\exp(-ct)\frac{ac}{1-\alpha}(\int_0^t \exp(ct)dt-\int_0^t \exp(ct-b(1-\alpha)t)dt)$$
$$=\frac{ac}{1-\alpha}\exp(-ct)(\frac{1}{c}\exp(ct)|_0^t-\int_0^t \exp((c-b(1-\alpha))t)dt)$$
$$=\frac{ac}{1-\alpha}\exp(-ct)(\frac{1}{c}(\exp(ct)-1)-\frac{\exp((c-b(1-\alpha))t)}{c-b(1-\alpha)}|_0^t)$$
$$=\frac{ac}{1-\alpha}\exp(-ct)(\frac{1}{c}(\exp(ct)-1)-\frac{\exp((c-b(1-\alpha))t)-1}{c-b(1-\alpha)})$$
$$=\frac{ac}{1-\alpha}(\frac{1}{c}-\frac{\exp(-ct)}{c}-\frac{\exp(-b(1-\alpha)t)-\exp(-ct)}{c-b(1-\alpha)})$$

(17)

因此，公式（14）和（17）分别是故障检测和故障排错的软件可靠性模型。

2.4 模型参数估计方法

本文所采用的参数估计方法是最小二乘估计（LSE）。由于故障排错过程依赖于故障检测过程，因此在参数估计中必须同时考虑故障检测模型和故障排错模型。因此，考虑到故障检测和排错过程[16]，可以使用下面的公式来估计模型参数。

$$S=\sum_{i=1}^{k}[(m_d(t_i)-m_{t_i})^2+(m_c(t_i)-n_{t_i})^2] \tag{18}$$

上式对提出模型参数变量求偏微分得，

$$\frac{\partial S}{\partial a}=\frac{\partial S}{\partial b}=\frac{\partial S}{\partial c}=\frac{\partial S}{\partial \alpha}=0 \tag{19}$$

联立解上面的微分方程可求出提出模型的参数值。

3 模型拟合和预测性能比较

在本节中，使用了两个开源软件的故障数据集来估计本文所用的模型的参数。此外，使用了四个模型比较标准来评估模型的拟合性能，即，MSE_d、MSE_c、R_d^2 和 R_c^2，还用了两个模型的比较标准来评估模型的预测性能，即，RE_d 和 RE_c。对提出的模型还进行了相应的参数敏感性分析，以便进一步考察哪些参数对提出的模型有重要的影响。

3.1 开源软件故障数据集

本文所使用的两个开源软件故障数据集来自文献[23]。它们是从在线 bug 跟踪系统 Bugzilla 中 Mozilla (https://bugzilla.mozilla.org/)和 GNOME(https://bugzilla.gnome.org/)两个开源软件项目收集和重新整理得到的。故障数据集之一是 Firefox 3.0、3.5 和 3.6，它们是三个连续版本，并且从 Appendix 列表中收集得到。另一个故障数据集是来自 GNOME 开源软件项目，包括四个连续的版本，它们也是从 Appendix 列表中收集得到的。例如，2.0x，2.1x，2.2x 和 2.3x。本文使用的故障数据集更详细信息，请参阅文献[23]。

3.2 模型比较标准

1）模型的拟合性能比较标准（Goodness-of-fit criteria）

$$MSE_d=\frac{\sum_{i=1}^{k}[m_d(t_i)-m_{t_i}]^2}{k} \tag{20}$$

$$MSE_c=\frac{\sum_{i=1}^{k}[m_c(t_i)-n_{t_i}]^2}{k} \tag{21}$$

其中，MSE_d 和 MSE_c 分别表示故障检测和故障排错过程中的均值平方误差。越小的 MSE_d 和 MSE_c 值，表示模型拟合性能越好。

$$R_d^2=1-\frac{\sum_{i=1}^{k}(m_d(t_i)-m_{t_i})^2}{\sum_{i=1}^{k}(m_{t_i}-\bar{m}_{t_i})^2},\ \bar{m}_{t_i}=\frac{1}{k}\sum_{i=1}^{k}m_{t_i} \tag{22}$$

$$R_c^2=1-\frac{\sum_{i=1}^{k}(m_c(t_i)-n_{t_i})^2}{\sum_{i=1}^{k}(n_{t_i}-\bar{n}_{t_i})^2},\ \bar{n}_{t_i}=\frac{1}{k}\sum_{i=1}^{k}n_{t_i} \tag{23}$$

其中，R_d^2 和 R_c^2 被用来评测故障检测和故障排错过程中模型的拟合性能。R_d^2 和 R_c^2 的值越接近于 1，模型的拟合性能越好。

a) 模型的预测性能比较标准，

b)

$$RE_d=\frac{m_d(t_{i+1})-m_{t_{i+1}}}{m_{t_{i+1}}} \tag{24}$$

$$RE_c=\frac{m_c(t_{i+1})-n_{t_{i+1}}}{n_{t_{i+1}}} \tag{25}$$

RE_d 和 RE_c 分别表示故障检测和故障排错的相对误差（the relative errors，RE）。相对误差是一步预测（one-step prediction），用于测量模型的预测性能。公式（24）和（25）分别表示在故障检测和故障排错过程中，到 t_i 时刻为止检测或去除故障数据来估计模型的参数值，并计算 t_{i+1} 时刻故障检测和故障排错的数量。RE_d 和 RE_c 越接近于零，表示模型的预测性能越好。当 RE_d 和 RE_c 大于零时，表示高估了软件中故障检测和故障去除的数量。否则，当 RE_d 和 RE_c 小于零时，表示低估了软件中故障检测和故障去除的数量。

3.3 模型性能比较和分析

3.3.1 模型拟合性能比较（Goodness-of-Fit）

从表 1，可以看到，在开源软件项目 Firefox 3.0、3.5 和 3.6 中提出模型的 MSE_d 逐渐减少。这表明提出模型的故障检测拟合性越来越好。但是，提出模型的 MSE_c 却在开源软件项目 Firefox 3.0、3.5 和 3.6 中首先是下降，然后是增长的趋势。这一结果指出故障排错过程是一个复杂的过程，受到许多主观因素的影响，即调试者调试能力、调试技巧和调试心理变化等。

因此，故障排错过程比故障检测过程有更大的波动性。此外，提出模型的 R_d^2 和 R_c^2 与 MSEd 和 MSEc 在模型的拟合变化上具有相同的变化趋势。

表 1　用 100%的开源故障数据集(Firfox)对提出模型的拟合性能进行比较

Version	Firefox 3.0	Firefox 3.5	Firefox 3.6
提出模型参数估计	a = 55.3675 b = 0.1107 c = 0.0268 α=0.0004	a = 515.0846 b = 0.0032 c = 0.0688 α=0.0002	a = 169.4776 b = 0.0154 c = 0.0445 α=0.0003
MSE_d	23.46	20.04	7.9
MSE_c	17.46	1.48	12.37
R_d^2	0.8671	0.8911	0.9087
R_c^2	0.8790	0.9927	0.8700

表 2　用 100%的开源故障数据集(Gnome2)对提出模型的拟合性能进行比较

Version	Gnome2 Control Center 2.0x	Gnome2 Control Center 2.1x	Gnome2 Control Center2.2x	Gnome2 Control Center2.3x
提出模型参数估 计	a= 50.4212 b = 0.1874 c = 0.0494 α=0.0111	a= 170.1584 b = 0.0276 c = 0.1608 α=0.0914	a= 186.554 b = 0.0257 c = 0.0050 α=0.2641	a= 246.9515 b = 0.0192 c = 0.0647 α=0.1082
MSE_d	52.12	25.17	3.09	2.62
MSE_c	87.25	42.55	21.66	17.15
R_d^2	0.8282	0.8741	0.9788	0.9745
R_c^2	0.5377	0.4753	0.2697	0.9023

从表 2 可以看到，提出模型的 MSE_d 在开源软件项目 Gnome2 Control-center 2.0x、2.1x、2.2x 和 2.3 中逐渐降低。提出模型的 MSE_c 在开源软件项目 Gnome2 Control-center 2.0x、2.1x、2.2x 和 2.3 中也是逐渐减少。另外，提出模型的 R_d^2 基本上是逐渐增加的。而且，提出模型的 R_c^2 在开源软件项目 Gnome2 Control-center 2.0x、2.1x、2.2x 到 2.3 却是首先减少，然后增加。从 R_c^2 变化情况看，能够得出在开源软件项目 Gnome2 Control-center 中排错过程仍是一个复杂和不规律变化的情况。例如，提出模型的 R_c^2 在开源软件项目 Gnome2 Control-center 2.2x 的值较低，也说明开源软件排错过程的不确定性。

从表 1 和 2 也能够看到提出模型的 MSE_d 在开源软件项目 Firefox 和提出模型的 MSE_d 在开源软件项目 Gnome 的变化趋势一致，都是逐渐下降的。但是提出模型的 MSE_c 在开源软件项目 Firefox 和提出模型的 MSE_c 在开源软件项目 Gnome 的变化趋势不一致，一个是先减少后增加；一个是逐渐减少。此外，提出模型的 R_d^2 在开源软件项目 Firefox 和提出模型的 R_d^2 在开源软件项目 Gnome 的变化趋势一致。但是提出模型的 R_c^2 在开源软件项目 Firefox 和提出模型的 R_c^2 在开源软件项目 Gnome 的变化趋势相反。一个是先增加后减少，一个是先减少后增加。

从图 1 也能看到提出模型的故障检测和故障排除的数量随测试时间的变化情况。图 1（a）和图 1（b）和（c）相比较，提出模型的故障检测和故障排错过程的拟合性能略差。图 1（d）与图 1（e）、（g）相比较，提出的模型的故障检测和故障排错过程的拟合性能较差。

从上面的分析，可以得出下面几点，

a) 提出模型的故障检测过程的拟合性能要好于提出模型的故障排错过程的拟合性能。

b) 在开源软件早期的版本中，提出模型的故障检测和故障排错的拟合性能相当，都很一般。

c) 在开源软件开发和测试过程中，故障排错过程比故障检测过程更加困难、复杂和不确定性。

d) 一般来说，提出的模型在整个开源软件故障数据集（包括故障检测和故障排错故障数据集）上有较好的拟合性能。

3.3.2 模型预测性能比较

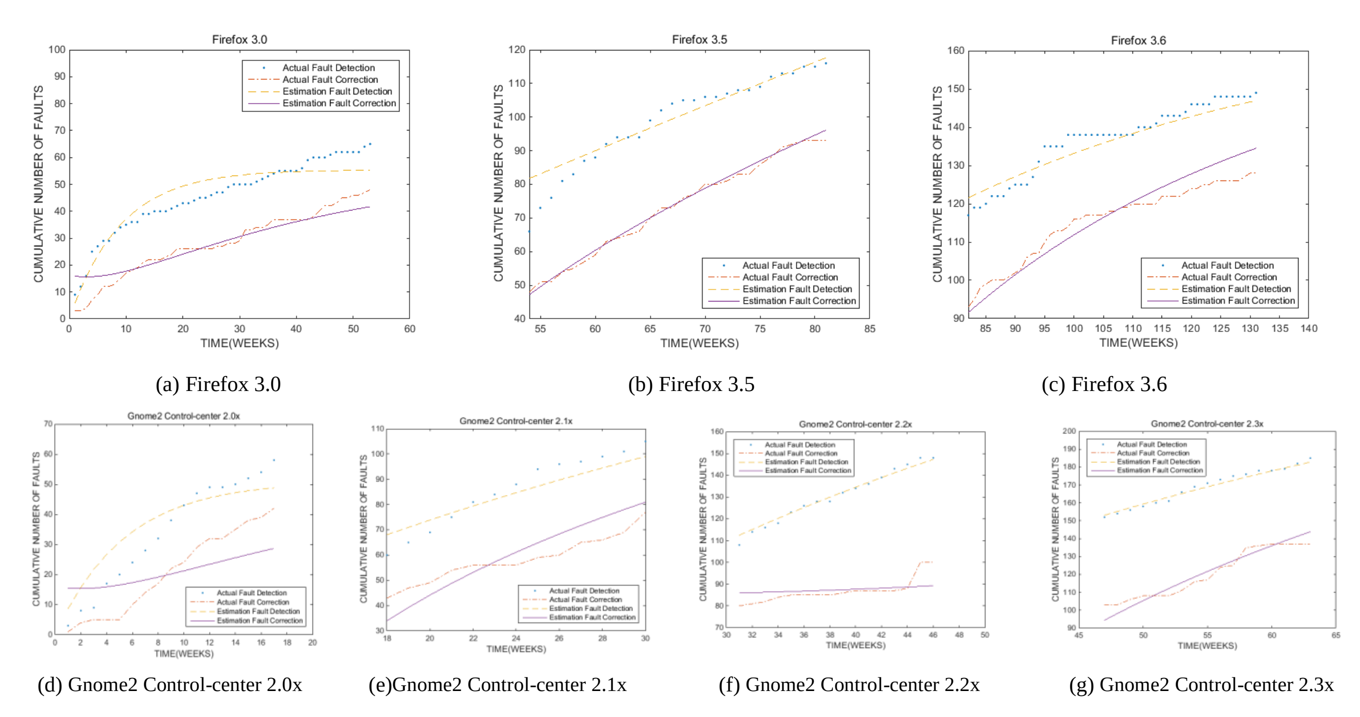
从图 2 中可以看出，提出的模型在开源软件中期版本中的故障排错过程比故障检测过程具有更好的预测性能。但提出的模型在开源软件后期版本中的故障检测过程比故障排错过程具有更高的预测精度。这说明故障排错过程比故障检测过程更加复杂和多变。一般来说，从图 2 可以看到，提出的模型在故障检测和故障排错过程中都有很好的预测性能。

从图 3 可以看到，提出的模型在整个故障数据集上故障检测过程比故障排错过程具有更好的预测性能。它表明预测排错故障的数量比预测检测到的故障数量要困难得多。此外，从图 3 还可以看到，提出的模型在故障检测和故障排错上有良好预测未来的失效和修正行为。

3.3.3 提出模型的敏感性分析

敏感性分析是确定模型中哪些参数有重要影响的一种方法。敏感性分析的一般方法是改变模型中一个参数的值并让模型中其它参数值保持不变[25]。

从图 4 中可以清楚地看到提出模型中参数 a、α 和 c 有重要的影响。直觉上来讲，提出的模型中参数 α 与开源软件在开发和测试过程中的故障引入有关。提出的模型中参数 c 与开源软件在开发和测试过程中的故障排错现象有关。提出的模型中参数 a 与开源软件在开发和测试过程中的初始故障总数有关。它们都与开源软件在开发和测试过程中影响软件可靠性建模的重要因素有关，如引入故障的数量、故障检测和故障排错之间的时间延迟以及开源软件中的初始故障总数。因此，敏感性分析的结果与开源软件的实际测试情况是一致的。故障引入和故障检测与排错之间的时间延迟是建立开源软件可靠性模型的两个重要因素。



(a) Firefox 3.0 (b) Firefox 3.5 (c) Firefox 3.6

(d) Gnome2 Control-center 2.0x (e)Gnome2 Control-center 2.1x (f) Gnome2 Control-center 2.2x (g) Gnome2 Control-center 2.3x

图 1　提出模型估计累计检测和排错故障数量与实际观察到的故障检测和排错的数量进行比较情况

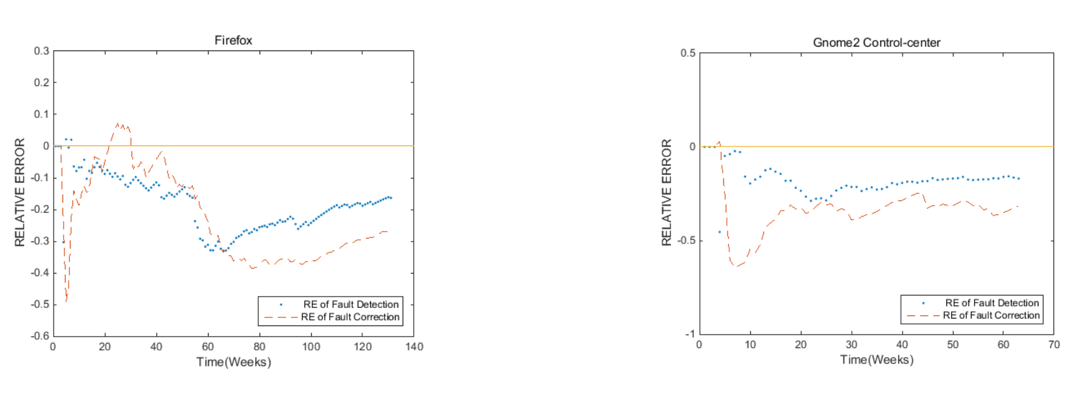
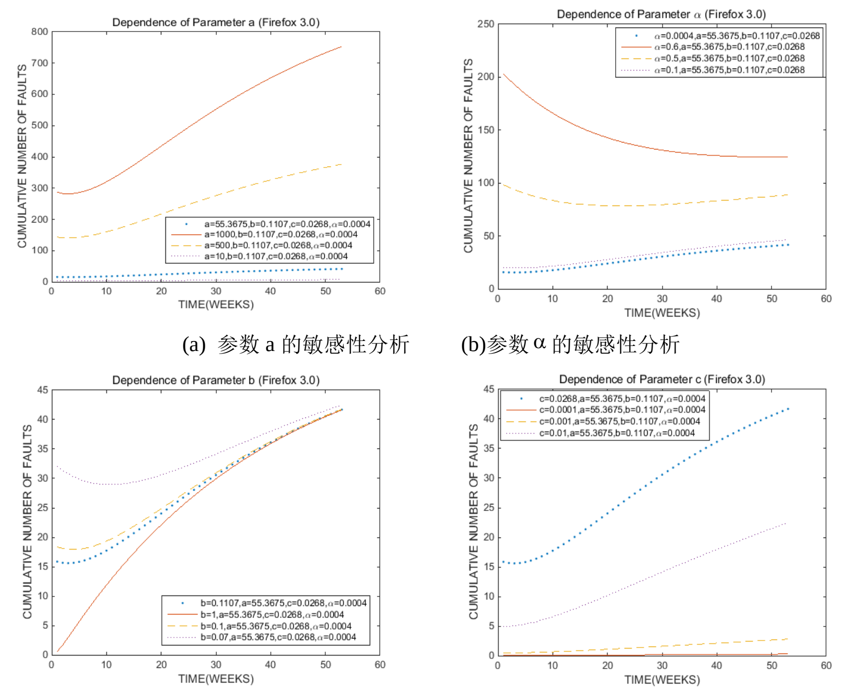


图 2　提出模型用开源软件项目 Firefox 作为故障数据集情况下的故障检测和排错过程的相对误差比较

图 3　提出模型用开源软件项目 Gnome2 Control-center 作为故障数据集情况下的故障检测和排错过程的相对误差比较



(a) 参数 a 的敏感性分析 (b)参数 α 的敏感性分析

(c)参数 b 的敏感性分析 (d)参数 c 的敏感性分析

图 4　提出的故障排错模型的参数敏感性分析

4 结束语

本文提出了一种整合故障排错和故障引进过程的开源软件可靠性模型。特别是对故障引入现象进行了相应的研究，并首次将其整合到开源软件的排错过程中。使用了两个真实的开源软件故障数据集来验证模型的拟合性能和预测性能。实验结果表明，提出的模型具有良好的拟合和预测性能，可以应用于开源软件的可靠性测试过程中。本文还讨论并分析了提出模型的参数敏感性。结果表明参数 a、c 和 α 对模型有重要影响。而且故障的引入和故障检测与排错之间的时间延迟是两个重要因素，建立开源软件可靠性模型时需要慎重考虑这两个重要因素。

虽然提出的模型可以有效地评估某些环境下的开源软件的可靠性，但还需要做深入的研究。例如，剩余的未修正的排错率随着测试时间不规则波动和故障引进的非线性变化等，这些现象是开源软件可靠性建模未来需要研究的方向。

参考文献：

[1] Raymond E S. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary [J]. 1999 (2): 105-107.

[2] Farber D. Six barriers to open source adoptione [R/OL] , (2004-03) . http://techupdate. zdnet. com/techupdate/stories/main/.

[3] Goel A L, Okumoto K. Time-dependent error-detection rate model for software reliability and other performance measures [J]. IEEE Trans on Reliability. 1979 (R-28): 206-211.

[4] Yamada S, Ohba M, Osaki S. S-shaped reliability growth modeling for software error detection [J]. IEEE Trans on Reliability. 1983 (32): 475-484.

[5] M. Ohba. Inflection S-shaped software reliability growth models. Stochastic Models in Reliability Theory [M]. Berlin Heidelberg: Springer, 1984: 144-162.

[6] Goel A L. Software Reliability Models Assumptions, Limitations and Applicability [J]. IEEE Trans on Software Engineering. 1985, 11 (12): 1411-1425.

[7] Pham H, Nordmann L, Zhang Xuemei. A general imperfect software debugging model with S-shaped fault detection rate [J]. IEEE Trans On Reliability. 1999, 48 (2): 169–175.

[8] Zhang Xuemei, Teng Xiaolin, Pham H. Considering fault removal efficiency in software reliability assessment [J]. IEEE Trans on Systems, Man, and Cybernetics—Part A: Systems and Humans. 2003, 33 (1): 2241–2252.

[9] Wang Jinyong, Wu Zhibo, Shu Yanjun, *et al.* An Imperfect Software Debugging Model Considering Log-logistic Distribution Fault Content Function [J]. Journal of Systems and Software. 2015, 100 (c): 167-181.

[10] Wang Jinyong, Wu zhibo, Shu Yanjun, *et al.* An Optimized Method for Software Reliability Model Based on Nonhomogeneous Poisson Process [J]. Applied Mathematical Modelling. 2016, 40 (13-14): 6324-6339.

[11] Schneidewind N F. Modelling the fault correction process [C]// . Proc of the 12th International Symposium on Software Reliability Engineering. Washington, DC, USA: IEEE Computer Society, 2001: 185–190.

[12] Xie Min. Zhao Min. The Schneidewind software reliability model revisited [C]// . Proc of the 3rd International Symposium on Software Reliability Engineering. Washington, DC, USA: IEEE Computer Society, 1992: 184–192.

[13] Lo Jung-Hua, Huang Chin-Yu. An integration of fault detection and correction processes in software reliability analysis [J]. Journal of Systems & Software. 2006, 79 (9): 1312-1323.

[14] Huang Chin-Yu, Lin Chu-Ti. Software reliability analysis by considering fault dependency and debugging time lag Reliability [J]. IEEE Trans On Reliability. 2006, 55 (3): 436-450.

[15] Wu Y P, Hu Q P, Xie Min, *et al.* Modeling and analysis of software fault detection and correction process by considering time dependency [J]. IEEE Trans on Reliability. 2007, 56 (4): 629–642.

[16] Xie Min, Hu Q P, Wu Y P, *et al.* A study of the modeling and analysis of software fault-detection and fault-correction processes [J]. Quality & Reliability Engineering. 2007, 23 (4): 459–470.

[17] Liu Yu, Li Duo, Wang Lujia, *et al.* A general modeling and analysis framework for software fault detection and correction process [J]. Software Testing Verification & Reliability, 2016, 26 (5) 351-365.

[18] Tamura Y, Yamada S. Optimisation analysis for reliability assessment based on stochastic differential equation modelling for open source software [J]. International Journal of Systems Science. 2009, 40 (4): 429–438.

[19] Zhou Ying, Davis Joseph. Open source software reliability model: an empirical approach [C]// . Proc of the 15th Workshop on Open Source Software Engineering, New York: ACM Press, 2005: 1–6.

[20] Li Xiang, Li Yan-Fu, Xie Min, *et al.* Reliability analysis and optimal version-updating for open source software [J]. Information and Software Technology, . 2011, 53 (9): 929–936.

[21] Peng R, Li Y F, Zhang W J, *et al.* Testing effort dependent software reliability model for imperfect debugging process considering both detection and correction [J]. Reliability Engineering & System Safety. 2014, 126 (2): 37–43.

[22] Liu Yu, Xie Min, Yang Jianfeng, *et al.* A new framework and application of software reliability estimation based on fault detection and correction processes [C]// . IEEE International Conference on Software Quality, Reliability and Security. Washington, DC, USA: IEEE Computer Society, 2015: 65-74.

[23] Yang Jianfeng, Liu Yu. Xie Min, *et al.* Modeling and analysis of reliability of multi-release open source software incorporating both fault detection and correction processes [J]. Journal of Systems & Software. 2016, 115 (c): 102-110.

[24] Ullah N, Morisio M. Vetro A. Selecting the best re-liability model to predict residual defects in open source software [J]. Computer. 2015, 48 (6):

50–58.

[25] Li Xiang, Xie Min, Ng S H. Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points [J]. Applied Mathematical Modelling 2010, 34 (11): 3560–3570.